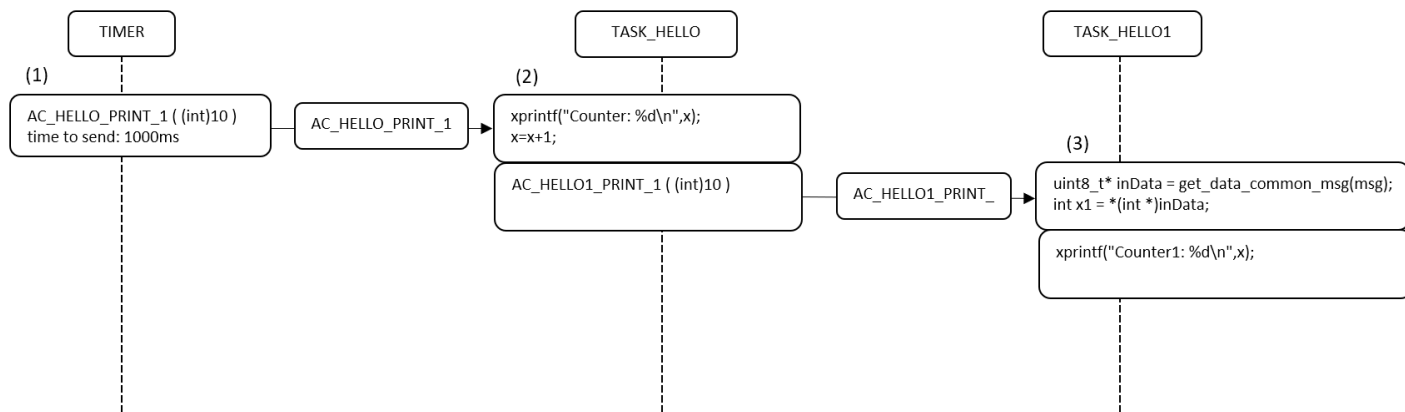


Hướng dẫn tạo task, set timer, gửi message và nhận message.

1) Trình tự kết nối:



Hình 1.1: Sơ đồ trình tự kết nối các task

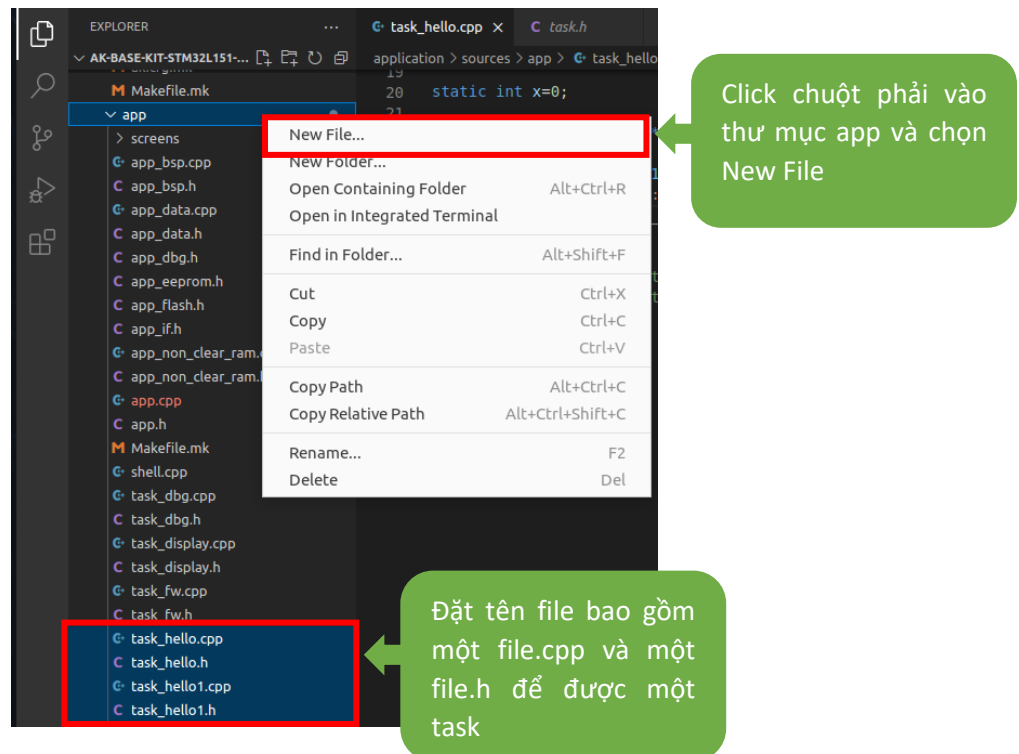
(1): Timer sẽ gửi tín hiệu đến TASK_HELLO một lần mỗi giây.

(2): Khi nhận được tín hiệu từ timer TASK_HELLO sẽ in ra Counter: x và cộng biến x thêm 1 và gửi dữ liệu đến TASK_HELLO1.

(3): Có dữ liệu từ TASK_HELLO, TASK_HELLO1 sẽ lấy dữ liệu và in ra Counter1: x.

2) Tạo Task:

B1: Tạo task_hello và task_hello1:



B2: Khai báo các task vừa tạo trong task_list:

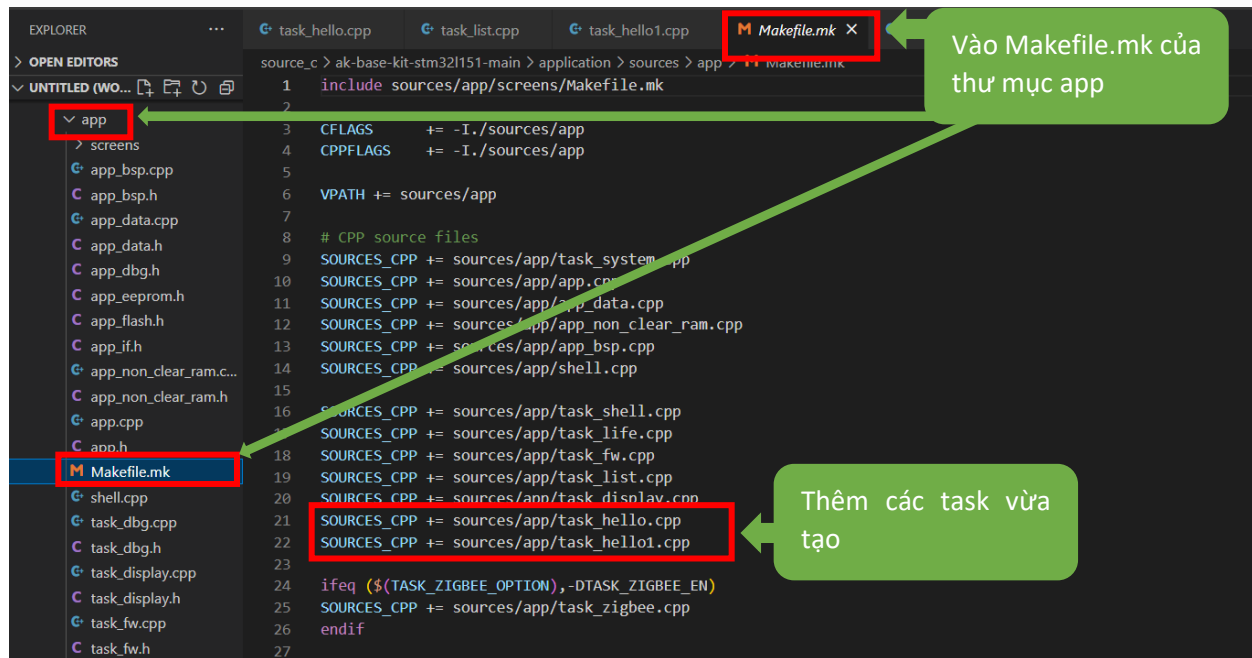
```
72  /*
73  /**
74  /* APP TASKS */
75  extern void task_system(ak_msg_t*);
76  extern void task_fw(ak_msg_t*);
77  extern void task_shell(ak_msg_t*);
78  extern void task_life(ak_msg_t*);
79  extern void task_if(ak_msg_t*);
80  extern void task_rf24_if(ak_msg_t*);
81  extern void task_uart_if(ak_msg_t*);
82  extern void task_dbg(ak_msg_t*);
83  extern void task_display(ak_msg_t*);
84  extern void task_zigbee(ak_msg_t*);
85
86  /* HELLO TASKS */
87  extern void task_hello(ak_msg_t*);
88  extern void task_hello1(ak_msg_t*);
```

Hình 2.3: Khai báo các task vừa tạo

```
5  /**
6  /* SYSTEM TASK */
7  /**
8  {TASK_TIMER_TICK_ID,      TASK_PRI_LEVEL_7,      task_timer_tick      },
9
10 /**
11 /* APP TASK */
12 /**
13 {AC_TASK_SYSTEM_ID      ,      TASK_PRI_LEVEL_2      ,      task_system      },
14 {AC_TASK_FW_ID          ,      TASK_PRI_LEVEL_2      ,      task_fw          },
15 {AC_TASK_SHELL_ID       ,      TASK_PRI_LEVEL_2      ,      task_shell       },
16 {AC_TASK_LIFE_ID        ,      TASK_PRI_LEVEL_6      ,      task_life        },
17 {AC_TASK_IF_ID          ,      TASK_PRI_LEVEL_4      ,      task_if          },
18 {AC_TASK_RF24_IF_ID     ,      TASK_PRI_LEVEL_4      ,      task_rf24_if     },
19 {AC_TASK_UART_IF_ID     ,      TASK_PRI_LEVEL_4      ,      task_uart_if     },
20 {AC_TASK_DBG_ID         ,      TASK_PRI_LEVEL_4      ,      task_dbg         },
21 {AC_TASK_DISPLAY_ID     ,      TASK_PRI_LEVEL_4      ,      task_display     },
22
23 /**
24 /* HELLO TASK */
25 /**
26 {AC_TASK_HELLO_ID       ,      TASK_PRI_LEVEL_4      ,      task_hello       },
27 {AC_TASK_HELLO1_ID      ,      TASK_PRI_LEVEL_4      ,      task_hello1      },
```

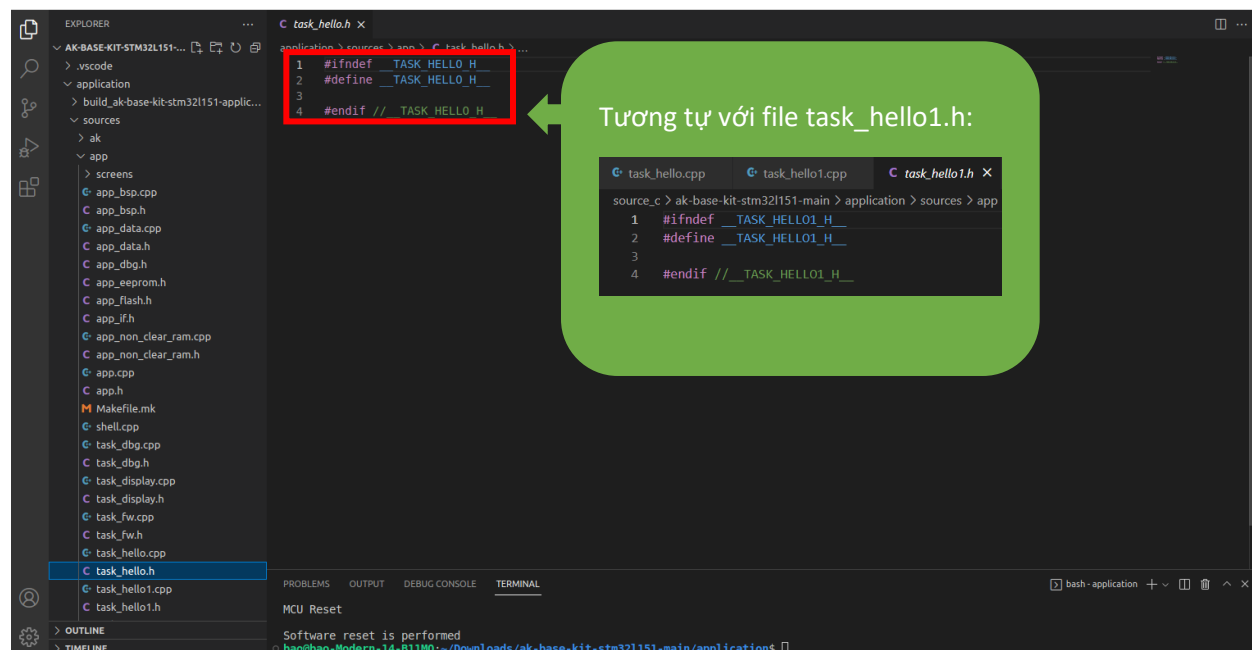
Hình 2.2: Set up TASK_PRI_LEVEL cho các task

B3: Cập nhật các task vừa tạo trong file make:

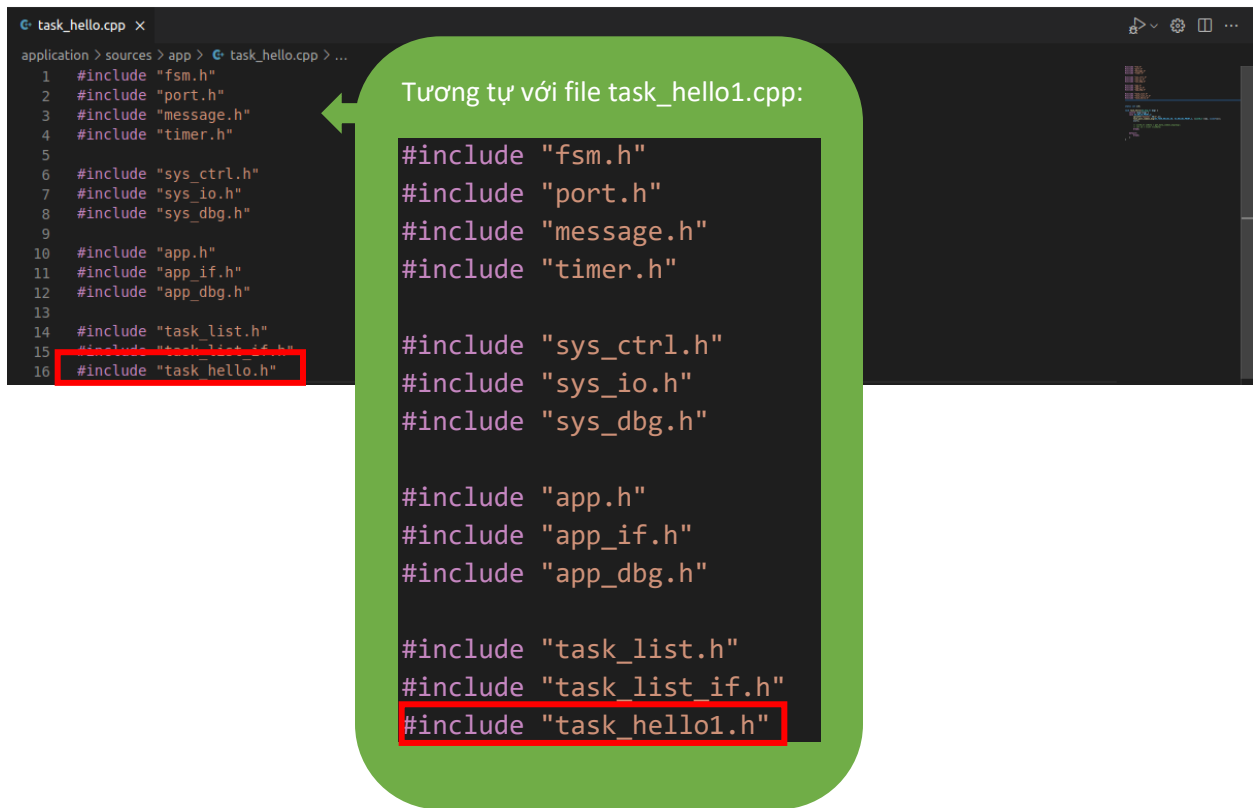


Hình 2.3: Cập nhật các task vừa tạo

B4: Set up các task:



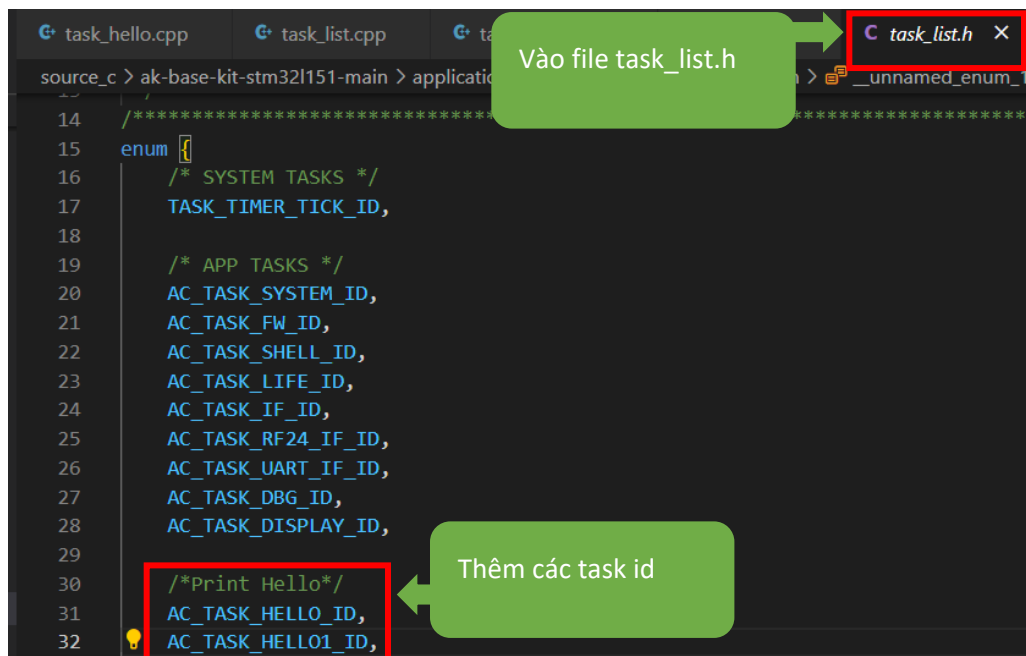
Hình 2.4: File task_hello.h



Hình 2.5: #include các thư viện của task_hello.cpp

3) Set timer

B1: Tạo task id:



Hình 3.1: Thêm các task id

B2: Tạo timer signal:

```
task_hello.cpp task_list.cpp task_h... app.h X
source_c > ak-base-kit-stm32l151-main > application >
181 AC_DBG_TEST_1 = AK_USER_DEFINE_SIG;
182 };
183
184 /**
185  * HELLO task define
186  */
187 /**
188  * define timer */
189  * define signal */
190 enum {
191 AC_HELLO_PRINT_1 = AK_USER_DEFINE_SIG,
192 };
193
194 enum {
195 AC_HELLO1_PRINT_1 = AK_USER_DEFINE_SIG,
196 };
197
```

Hình 3.2: timer signal

B3: Set timer

```
task_hello.cpp task_list.cpp task_h... app.cpp X app.h
source_c > ak-base-kit-stm32l151-main > application > app > app.cpp > ...
337 #endif
338 }
339 }
340
341 /**
342  * app initial function.
343  */
344 /**
345  *
346  */
347
348 are timer for application
349 tasks
350 timer() {
351 timer to toggle life led */
352 timer_set(AC_TASK_LIFE_ID, AC_LIFE_SYSTEM_CHECK, AC_LIFE_TASK_TIMER_LED_LIFE_INTERVAL, TIMER_PERIODIC);
353 timer_set(AC_TASK_HELLO_ID, AC_HELLO_PRINT_1, AC_TASK_TIMER_LED_LIFE_INTERVAL, TIMER_PERIODIC);
354 // timer_set(AC_TASK_HELLO1_ID, AC_HELLO1_PRINT_1, AC_TASK_TIMER_LED_LIFE_INTERVAL, TIMER_PERIODIC);
355 timer_set(AC_TASK_FW_ID, FW_CHECKING_REQ, FW_UPDATE_REQ_INTERVAL, TIMER_ONE_SHOT);
356 }

```

Hình 3.3: Set timer

B4: Hứng signal từ timer:

```
17 // #include "
18
19 #include "scr_noen.h"
20
21
22 static int x=0;
23
24 void task_hello(ak_msg_t* msg)
25 {
26     switch (msg->sig)
27     {
28         case AC_HELLO_PRINT_1:
29             x=x+1;
30             xprintf("Counter: %d\n",x);
31             task_post_common_msg(AC_TASK_HELLO1_ID, AC_HELLO1_PRINT_1, (uint8_t *)&x, sizeof(x));
32
33             // uint8_t* inData = get_data_common_msg(msg);
34             // int x2 = *(int *)inData;
35             break;
36
37         default:
38             break;
39     }
40 }
```

Vào file task_hello.cpp

Hứng signal từ timer

Sau đó xử lý các lệnh, hàm

Hình 3.4: Hứng signal từ timer

4) Gửi và nhận message

B1: Tạo task id và signal tương tự B1 và B2 của phần 3 để được task id và signal của TASK_HELLO1.

B2: Gửi message:

```
17 // #include "task_hello1.h"
18
19 #include "scr_noen.h"
20
21
22 static int x=0;
23
24 void task_hello(ak_msg_t* msg) {
25     switch (msg->sig) {
26     case AC_HELLO_PRINT_1:
27         x=x+1;
28         xprintf("Counter: %d\n",x);
29         task_post_common_msg(AC_TASK_HELLO1_ID, AC_HELLO1_PRINT_1, (uint8_t *)&x, sizeof(x));
30     }
31 }
32
33
34
35 task_post_pure_msg // Để gửi dữ liệu nhỏ như timer
36 task_post_common_msg // Để gửi dữ liệu với kích thước vừa
37 task_post_dynamic_msg // Để gửi dữ liệu big size
38
39
40 }
```

task_hello.cpp

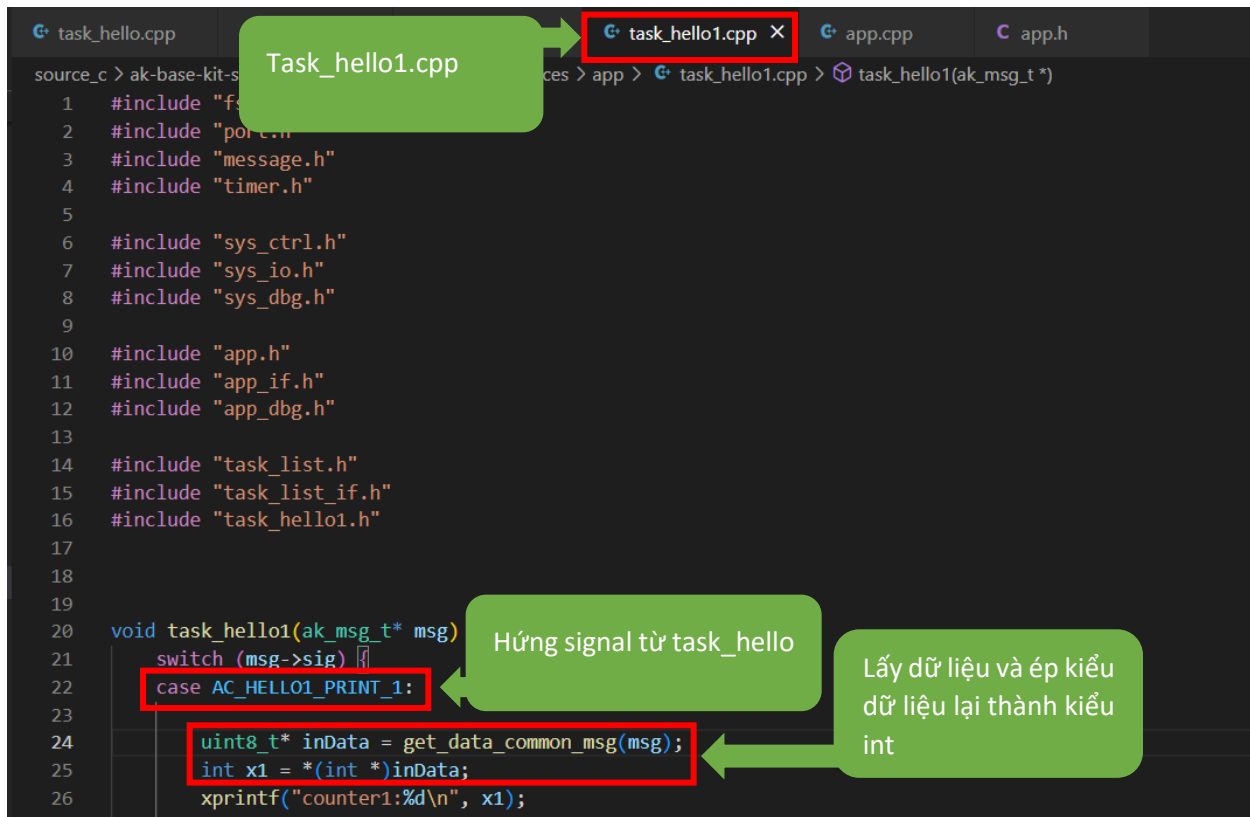
Ép kiểu dữ liệu cho biến x

Tùy thuộc vào kiểu dữ liệu muốn gửi đi có thể chọn các kiểu gửi khác nhau:

- task_post_pure_msg // Để gửi dữ liệu nhỏ như timer
- task_post_common_msg // Để gửi dữ liệu với kích thước vừa
- task_post_dynamic_msg // Để gửi dữ liệu big size

Hình 4.1: Gửi message

B3: Hứng message:



Hình 4.2: Hứng message

5) Kết quả:

Để kiểm tra kết quả gửi và nhận message giữa các task, ta vào minicom để xem:

```
311MO: ~/Downloads/ak-base-kit-stm3...
bao@ ~$ cd Do
bao@bao-Modern-14-B11M0: ~/Downloads/ak-base-kit-stm32l151-main/application$ sudo minicom -D /dev/ttyUSB0 -b 115200
[sudo] password for bao:
Welcome to mini...
OPTIONS: I18n
Port /dev/ttyUSB0, 15:27:02
Press CTRL-A Z for help on special keys
Counter: 44
counter2:44
Counter: 45
counter2:45
Counter: 46
counter2:46
Counter: 47
counter2:47
Counter: 48
```

Đường link dẫn đến thư mục application trên máy

Thực hiện cú pháp: `sudo minicom -D /dev/ttyUSB0 -b 115200` để vào minicom

Cứ mỗi giây task_hello nhận được tín hiệu từ timer và Counter sẽ đếm lên đồng thời gửi đến task_hello1 in ra Counter2